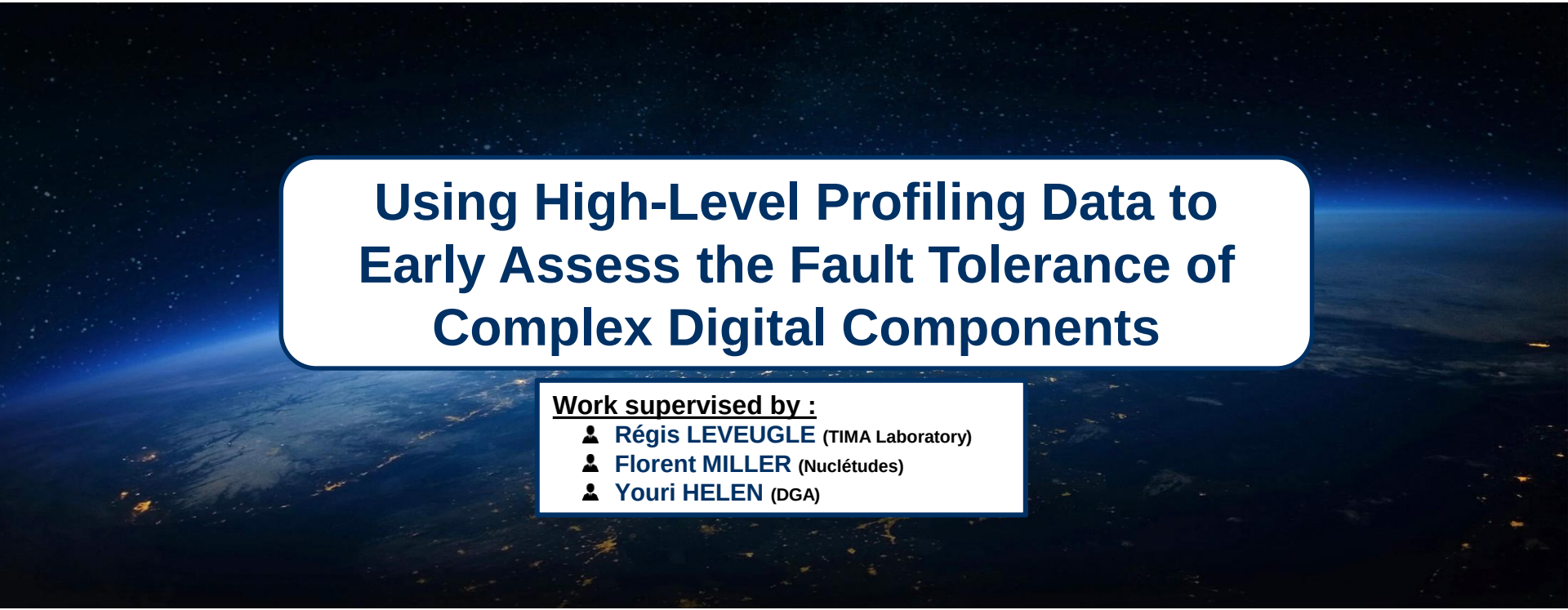


Using High-Level Profiling Data to Early Assess the Fault Tolerance of Complex Digital Components

Work supervised by :

- Régis LEVEUGLE (TIMA Laboratory)
- Florent MILLER (Nuclétudes)
- Youri HELEN (DGA)

Workshop SemSecuElec - Friday 13th November 2026

SUMMARY

1. **INTRODUCTION & MOTIVATIONS**
2. PROPOSED METHODOLOGY
3. CASE STUDY
4. EXPERIMENTAL RESULTS
5. CONCLUSION AND PERSPECTIVES

50 years of evolution of embedded processors

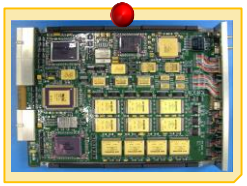
Yesterday - Today



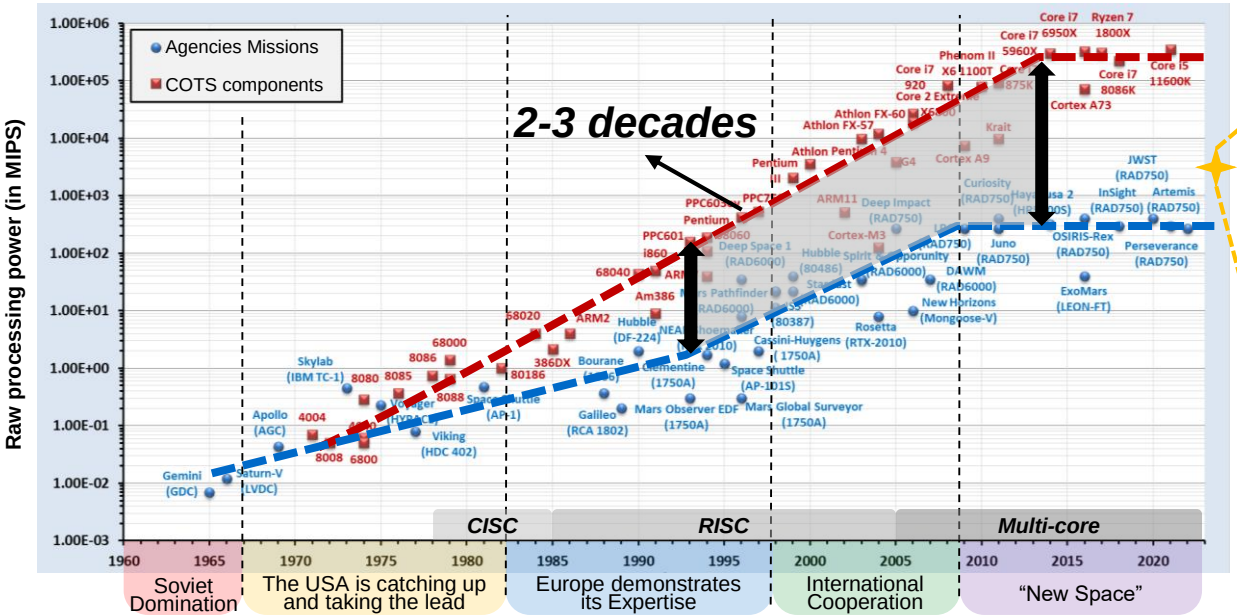
AGC - 1969



ERC32 - 2000



RAD750 - 2020



Tomorrow



GR765 - 2025

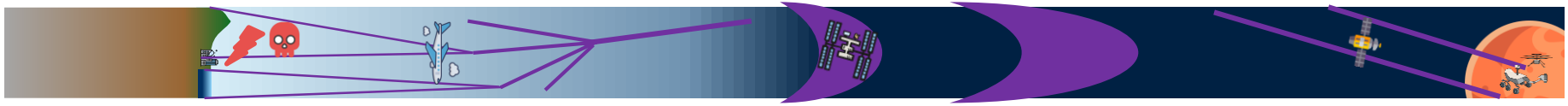
RISC-V



HPSC - 2025

CREACH LABS cybersecurity research

High Performance Embedded Computing (HPEC)



Ground level



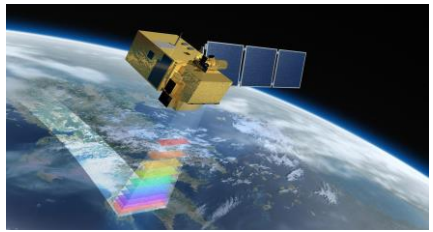
- Autonomous cars
- Biomedical
- IoT / Datacenters

Atmospheric



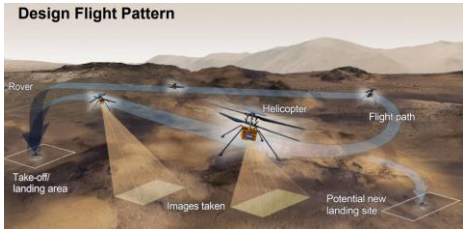
- Autonomous flights
- Flying processing
  - Crops analysis
  - Building 3D mapping

Earth Orbit



- Orbital Maneuvers
- In orbit processing
  - Ships detection
  - Environment analysis

Extra planetary

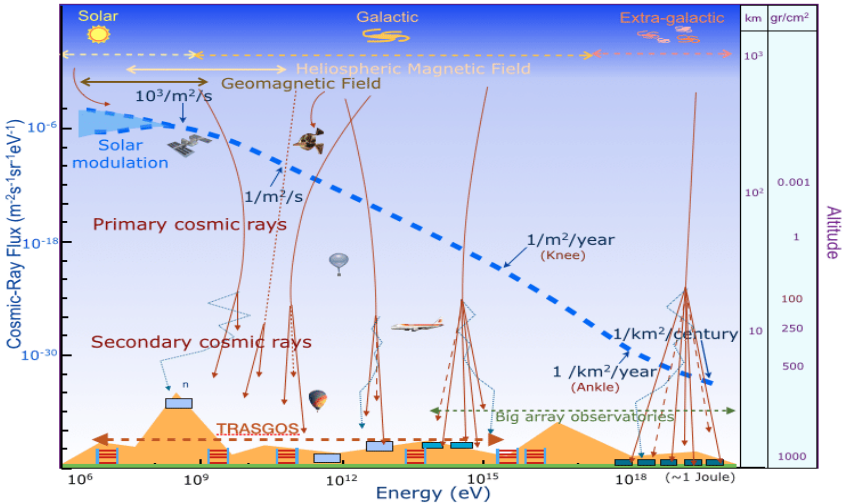


- Autonomous flights
- Autonomous rovers

Exposed to :
Malicious Attacks / Natural Radiation Effects

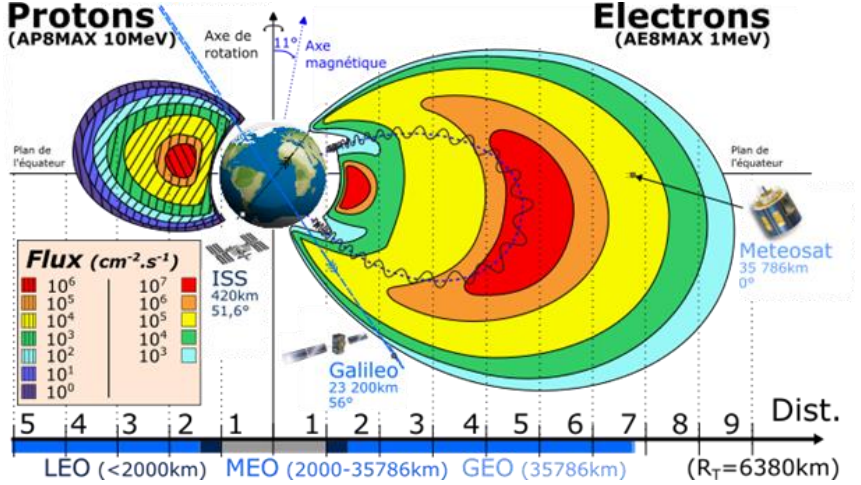
Natural Radiation Environment (NRE)

Atmospheric



[Garzon_2016]

Space

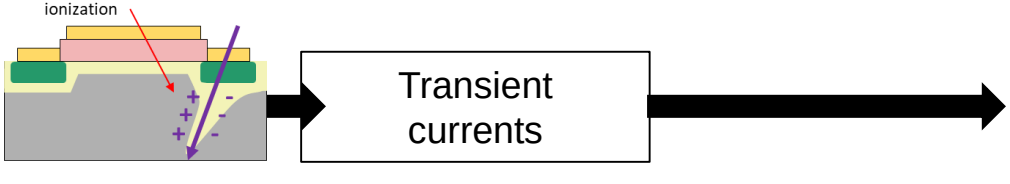


+ heavy ions coming from GCR (Galactic Cosmic Rays)

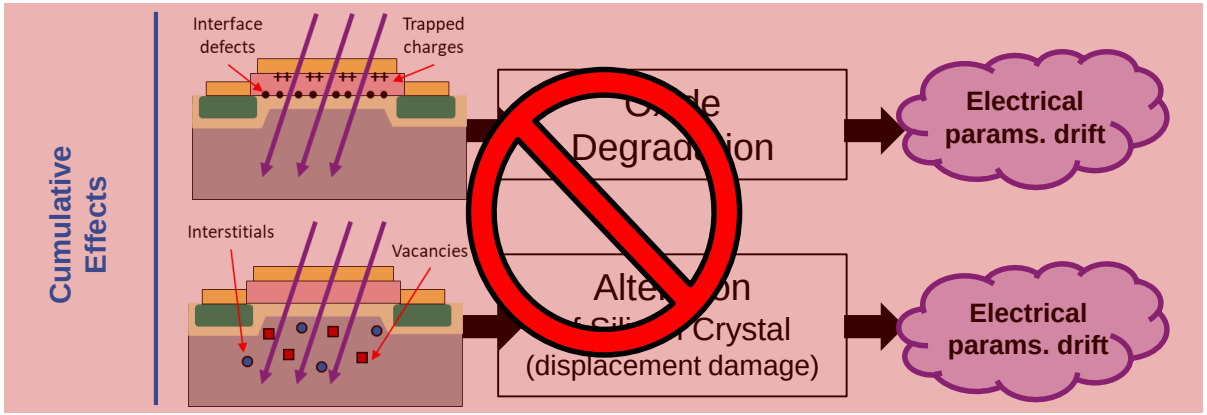
Question : How this natural environment impact complex digital components ?

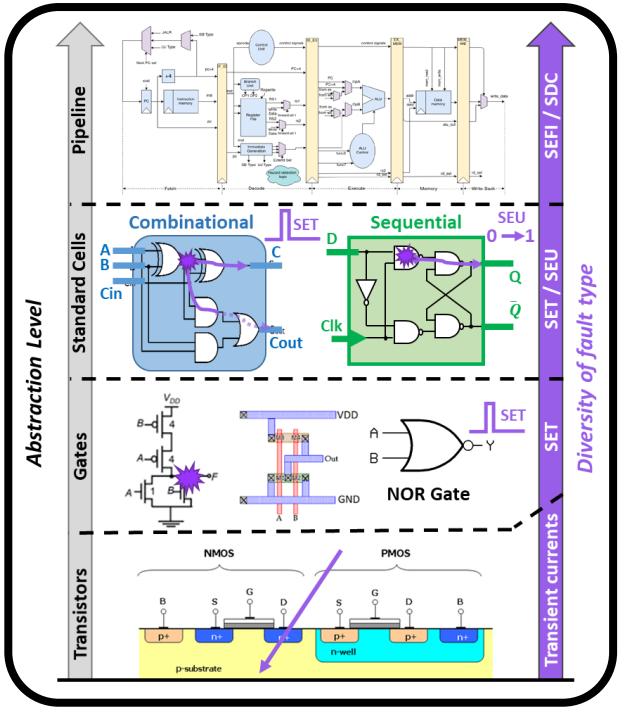
Radiation effects on digital complex components

Single Event Effects



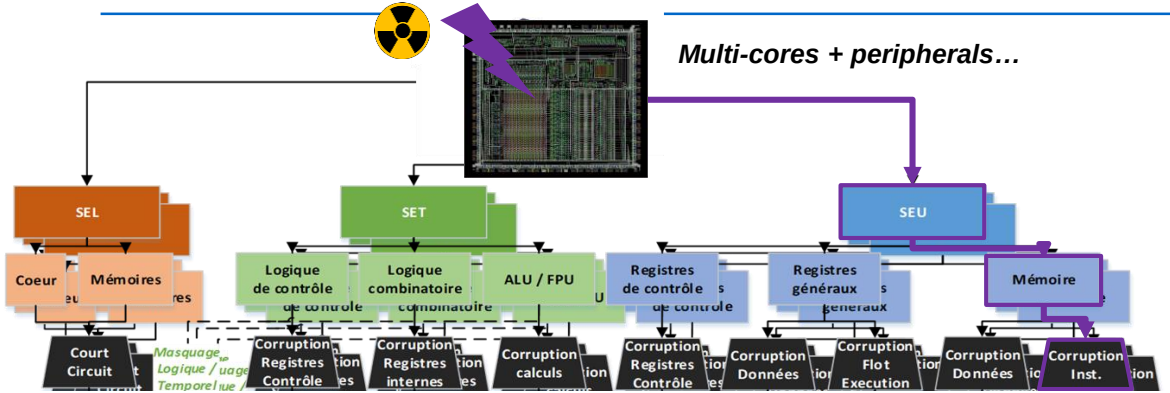
Cumulative Effects





Increase (at component level) with technological node

SoC Fault classification – Link between HW & SW



Example :

C code :

```
int function1(int in1, int in2) {
   return in1 + in2;
}
```

Assembly code :

```
0x80000000 : mv t0, a0
0x80000004 : mv t1, a1
0x80000008 : add t0, t1   sub t0, t1
0x8000000C : mv a0, t0
0x80000010 : ret
```

C code :

```
int function2(int in1, int in2) {
   if (in1 != 0) {return in1 + in2;}
   else {return 0;}
}
```

Assembly code :

```
0x80000000 : mv t0, a0
0x80000004 : mv t1, a1
0x80000008 : beqz t0, 80000018
0x8000000C : add t0, t1   sub t0, t1
0x80000010 : mv a0, t0
0x80000014 : ret
0x80000018 : mv a0, x0
0x8000001C : ret
```

Execution Flow / Execution flow

⚠ Silent Data Corruption

⚠ Crash or Hang

✓ No Impact

SEL = Single Event Latchup SET = Single Event Transient SEU = Single Event Upset (= bitflip)

Robustness of digital components is **COMPLEX** to estimate

Cross section – static and dynamic

Test Standards

ESCC-25100 – (ESA) Guide to assess the SEE rate of spatial components – **2014**

Metric to estimate the fault tolerance of a component :

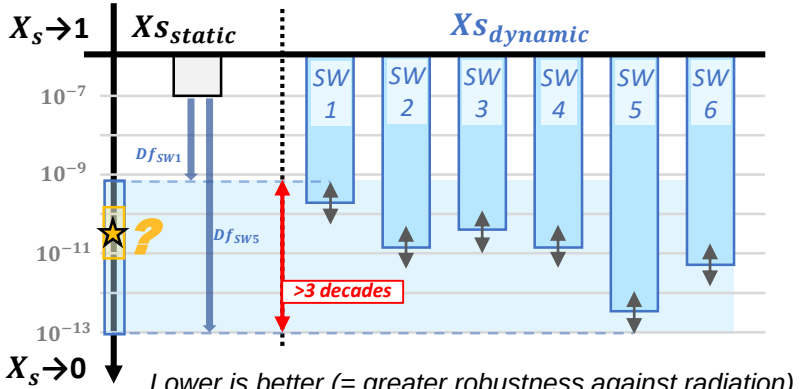
$$X_s = \frac{Nbr_{events}}{fluence}$$

(cm²/component)
(cm²/ressource)

Flux (n.cm^{-2}.s^{-1}) x Time (s)

Take into account the SW

Static Xs → Dynamic Xs
(applying a **Derating factor Df**)



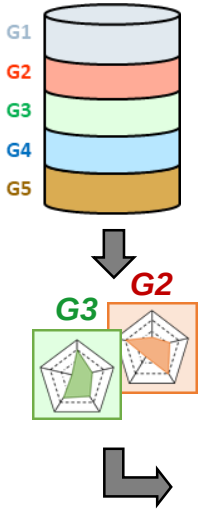
Lower is better (= greater robustness against radiation)

1) Estimate the robustness to radiation-induced errors of a complex component ⇔ Asses its Xs
2) Taking into account the SW ⇔ Calculate the appropriate derating factor (Df)

The SW dynamic component is not negligible

Question : How to reproduce the final SW at a time where is it not known / available ?

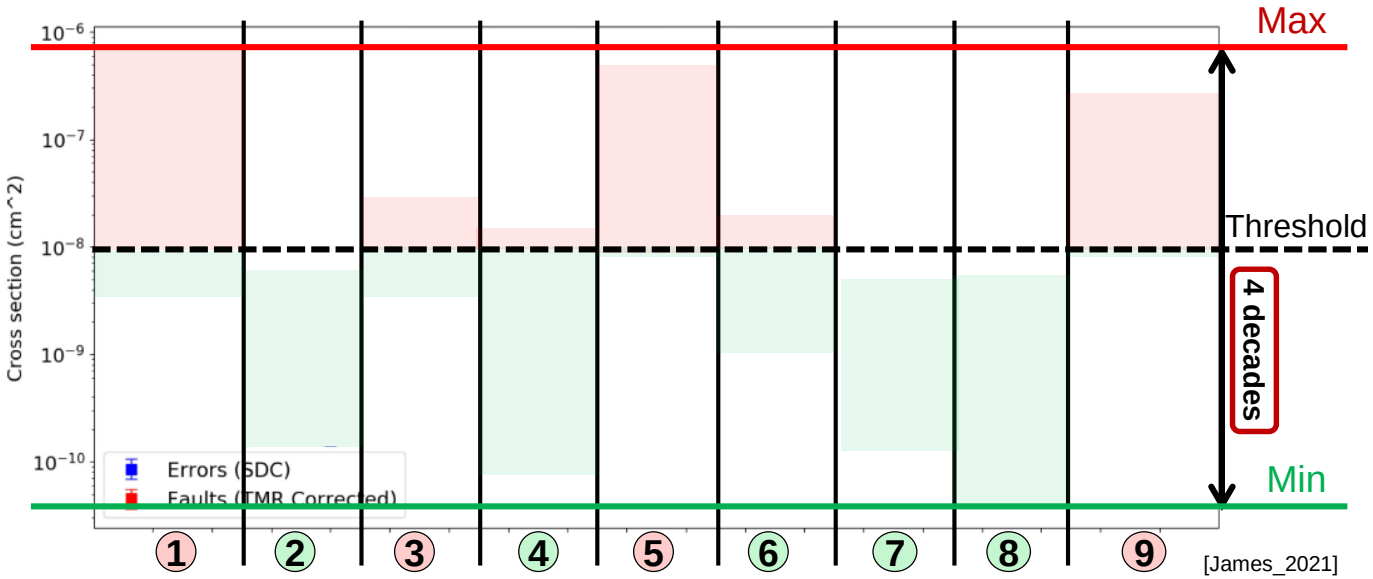
SW Database



Subset List

1. CRC32
2. Dijkstra
3. FFT
4. Mm_L2
5. MxM
6. Nanojpeg
7. Qsort_Lib
8. Qsort_Num
9. Sha256

SOFTWARE PROFILING

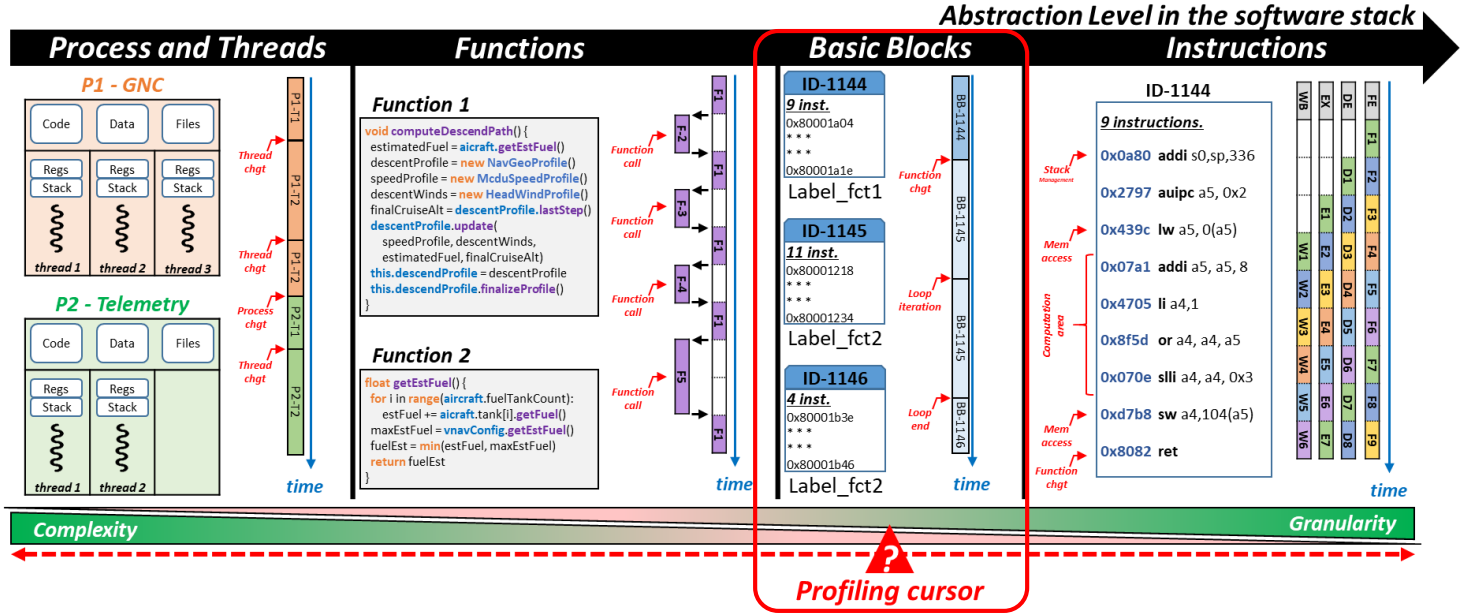


[James_2021]

Lower is better (= greater robustness against radiation)

CREACH LABS cybersecurity research

Which profiling level to choose ?



Selection of the « Basic Blocks » level for its modularity (level of abstraction & time granularity)

Which profiling metrics to choose ?

Type	Category	Name	Metric	Precision level of the metric		
				S	D	C
Block Description	Memory Localization	C-D.1	Block Address	NA	L	P
		C-D.2	Block Size	NA	M	P
	Traffic in the registers	C-D.3 C-D.6	Number of access and read rate per register	E	M	P
		C-D.7	Dependencies btw registers	NA	L	P
	Instructions mix	C-D.8 C-D.17	Breakdown of Inst. types	E	M	P
Transitions btw Blocks		C-T.1	% Forward branches (Fw)	NA	L	P
		C-T.2	% taken Fw branches	NA	L	P
Data access						

C-X.1	Number of access
C-X.2	Read rate
C-X.3	Mean Dist. btw access
C-X.4	Life time

S Specification D Development C Characterization

NA Not Available E Estimated L Low M Medium P Precise

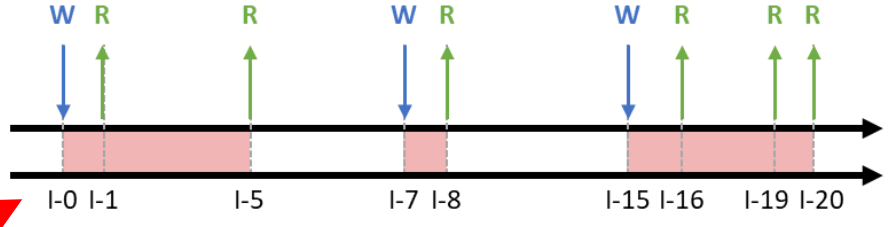
Detailed example :

C-X.1 – Nacc: 9

C-X.2 – Rrat: 6/9 = 0,67

C-X.3 – MDbwA: 20/9 = 2,22

C-X.4 – Lt: 11/20 = 0,55



State of the art metrics which are indpt. from the µArchi
→ 4 have been selected for the proof of concept

[Poovey_2009] [Joshi_2006] [Joshi_2008] [Sha_2018]

SUMMARY

1. INTRODUCTION & MOTIVATIONS

2. PROPOSED METHODOLOGY

3. CASE STUDY

4. EXPERIMENTAL RESULTS

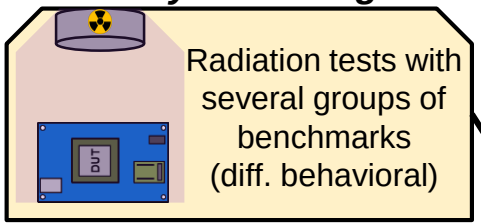
5. CONCLUSION AND PERSPECTIVES

Motivations of the work



Development Team — *Radiation Team*

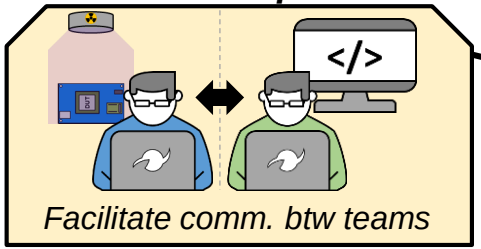
Early de-risking



Radiation tests with several groups of benchmarks (diff. behavioral)

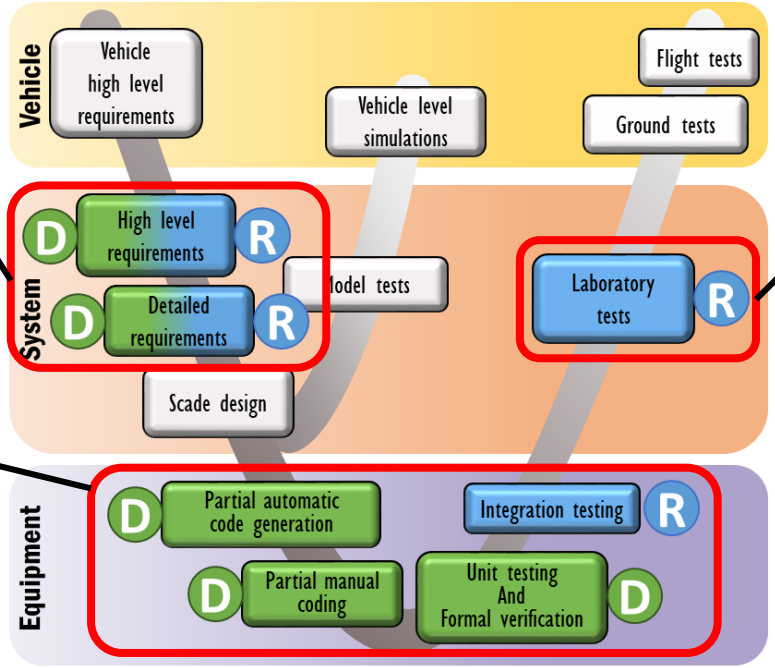
➡ **Select the digital component**

SW development

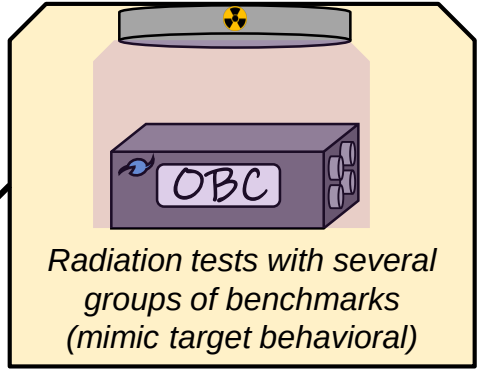


Facilitate comm. btw teams

➡ **Help for SW design choices**



Final characterization



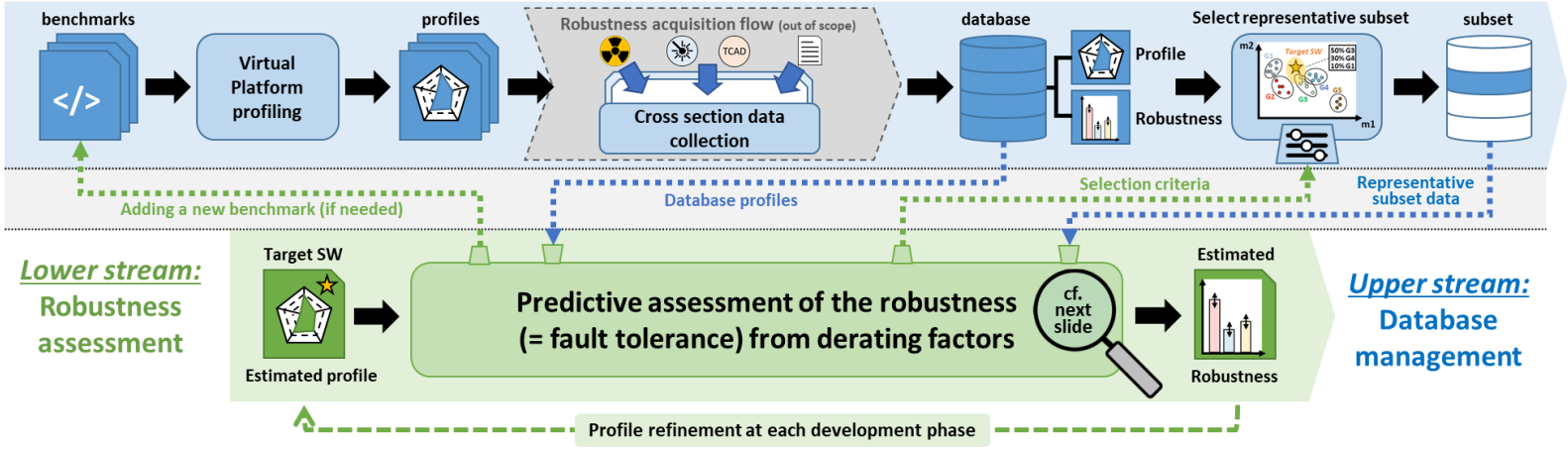
Radiation tests with several groups of benchmarks (mimic target behavioral)

➡ **Characterize the sel. component**

All along the dev. cycle without access to the final program

CREACH LABS cybersecurity research

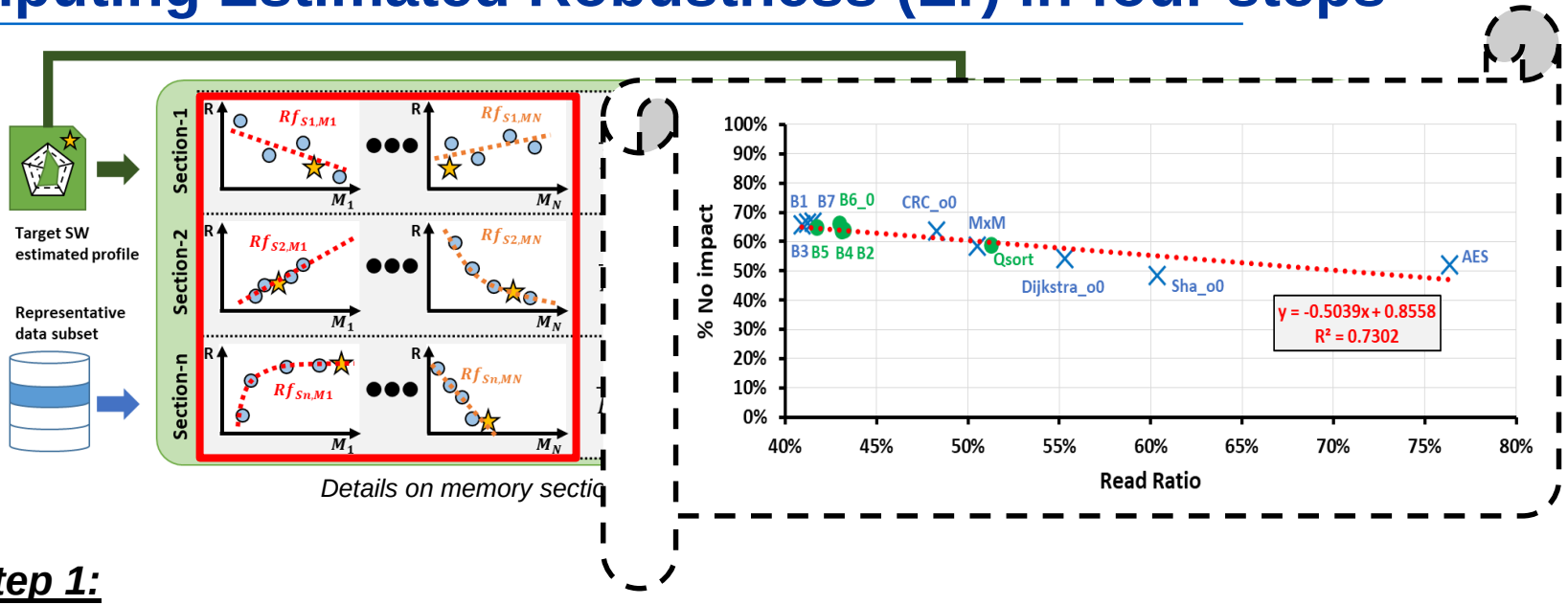
A 2-stream methodology



Main characteristics:

- Database management stream is independent
- Any instruction accurate virtual platform can be used
- Estimated fault tolerance can be refined during SW dev. phase
- Profiling metrics can be added / updated / removed

Computing Estimated Robustness (Er) in four steps

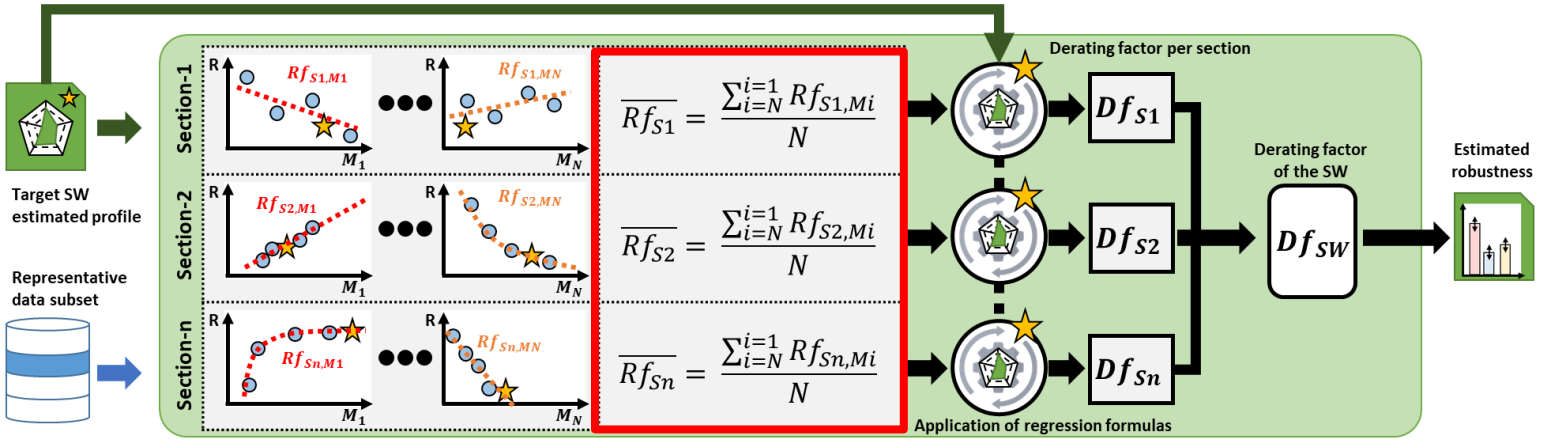


Step 1:

Identify regression formulas ($Rf_{Ss,Mi}$) for each section Ss and profiling metric Mi

R = Robustness, Mn = Profiling metric n

Computing Estimated Robustness (Er) in four steps



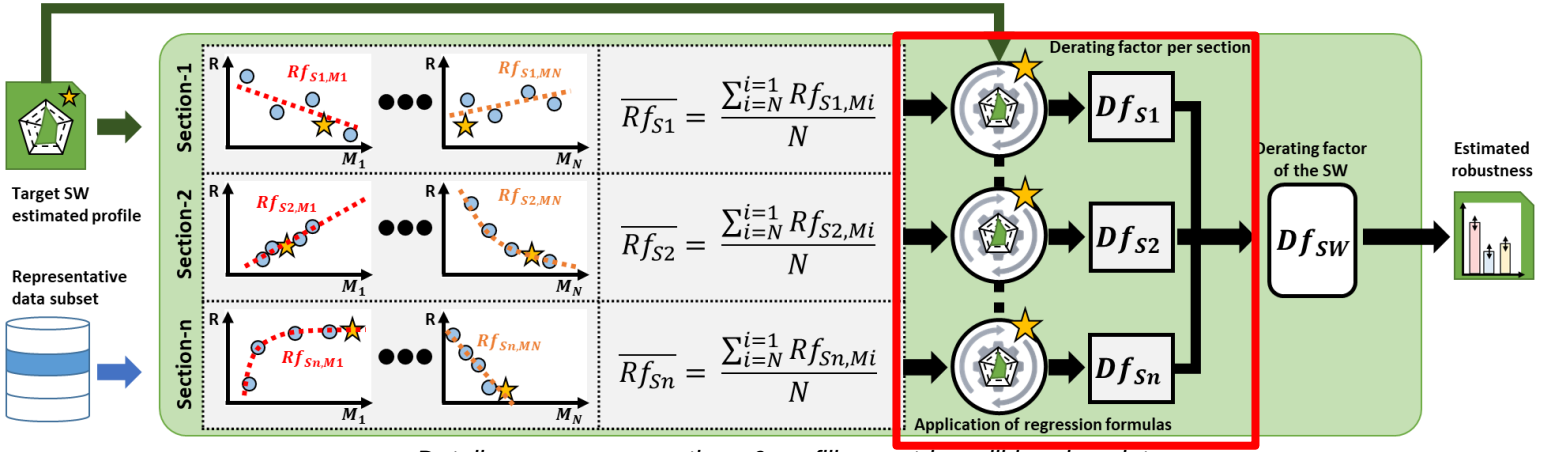
Details on memory sections & profiling metrics will be given later

Step 2:

Compute mean regression formula ($\overline{Rf_{Ss}}$) of each section s by averaging the regression formulas for all profiling metrics

R = Robustness, Mn = Profiling metric n

Computing Estimated Robustness (Er) in four steps



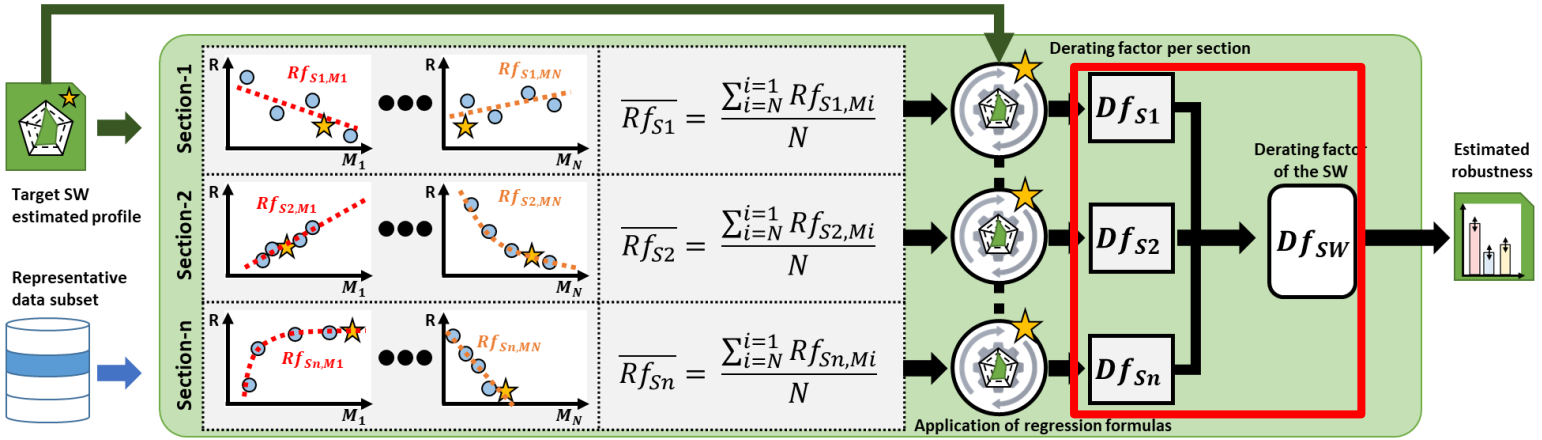
Details on memory sections & profiling metrics will be given later

Step 3:

Prev. computed $\overline{Rf_{Ss}}$ are applied with target estimated profile data to obtain derating factor Df_{Ss} associated to the section s

R = Robustness, Mn = Profiling metric n

Computing Estimated Robustness (Er) in four steps



Details on memory sections & profiling metrics will be given later

Step 4:

Apply these Df_{Ss} to their respective section to obtain the estimated robustness associated to the target software (noted Er_{SW})

$$Er_{SW} = \frac{\sum_s Df_s(SW_{profile}) \times N_s}{\sum_s N_s}$$

possible faults in the section s (@ selected fault model)

R = Robustness, Mn = Profiling metric n

SUMMARY

1. INTRODUCTION & MOTIVATIONS

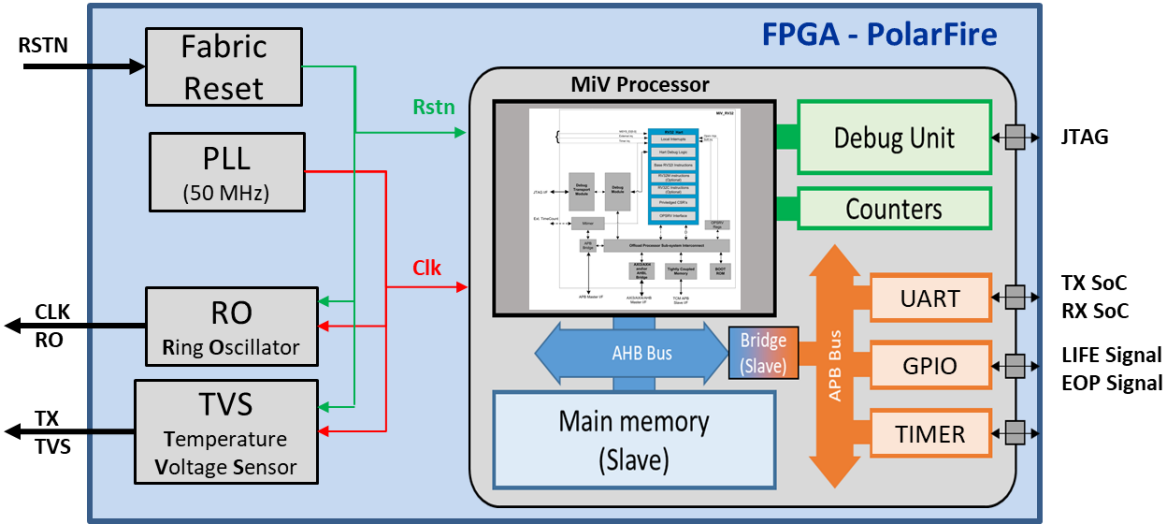
2. PROPOSED METHODOLOGY

3. CASE STUDY

4. EXPERIMENTAL RESULTS

5. CONCLUSION AND PERSPECTIVES

Device Under Test



Main characteristics :

- Soft-SoC RISC-V
- Instantiated on a Flash-based FPGA
- MiV core (4 stage pipeline)
- Bus AHB
- Multiple peripherals
- Ring oscillator
- Integrated TVS
  (Temperature and Voltage Sensor)

Perfect target for achieving faults injection & test under radiation beam

Softwares and observables

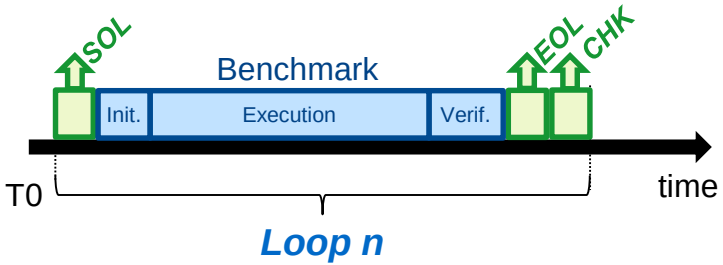
Benchmarks :

- 7 algorithms : Bitcount, Qsort, Dijkstra, CRC, MxM, SHA, AES
- 7 versions of Bitcount (B1-B7)
- Mainly from the Mibench
- 4 algorithms compiled with 3 different options (o0, o1, o3)

To reproduce the aspect of database / subset of the methodology
Some programs are used for « training » only

Fault classification :

Signed and Looped Execution

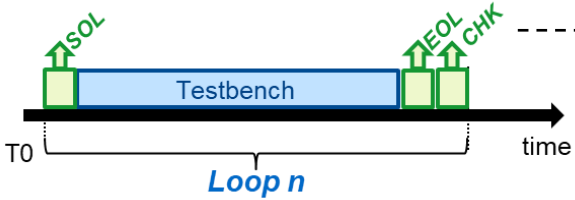


Use of simple and hardened communications

- Sequencing frame (SOL / EOL / CHK) **[UART]**
- TVS Data (Temp. & Voltage Sensor) **[UART]**
- Ring Oscillator frequency (dose drift) **[GPIO]**
- LIFE and EOL (End Of Loop) signals **[GPIO]**

Considered fault types

Marked looping with signed execution

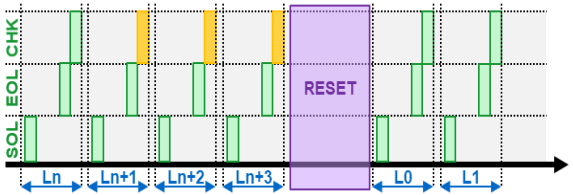


Checksum (ex : MxM result) is compared to a golden value (with no fault injection)

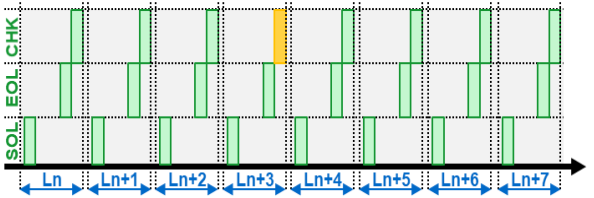
❖ **Type-1 Error: SEFI**



❖ **Type-2 Error: P-SDC**

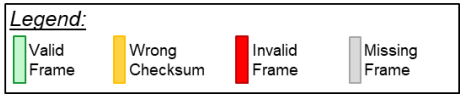


❖ **Type-3 Error: T-SDC**



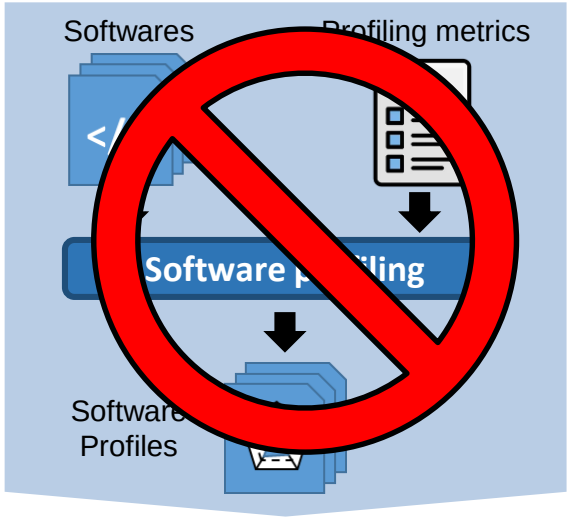
Measured Robustness :
(of a software)

$$Mr_{SW}(\%) = \frac{Nbr\ of\ faults\ having\ no\ impact}{Nbr\ of\ faults\ injected}$$



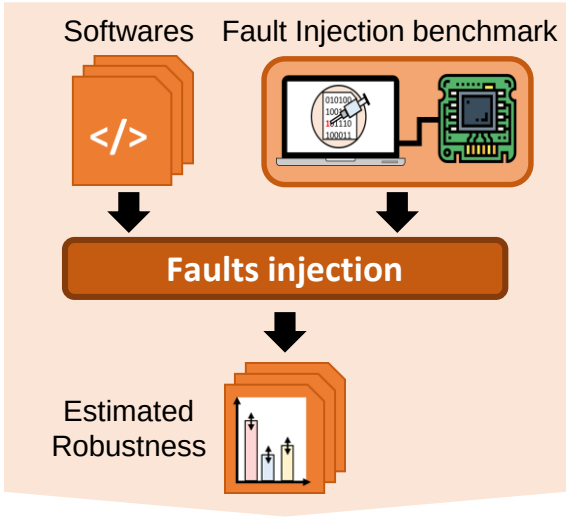
“Proof of concept” case study

Advanced profiling tool



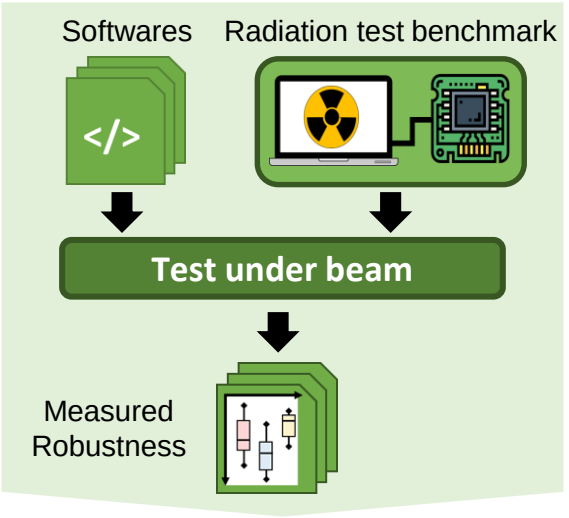
Versatile and detailed SW profiling
Not detailed in this presentation

Emulation platform



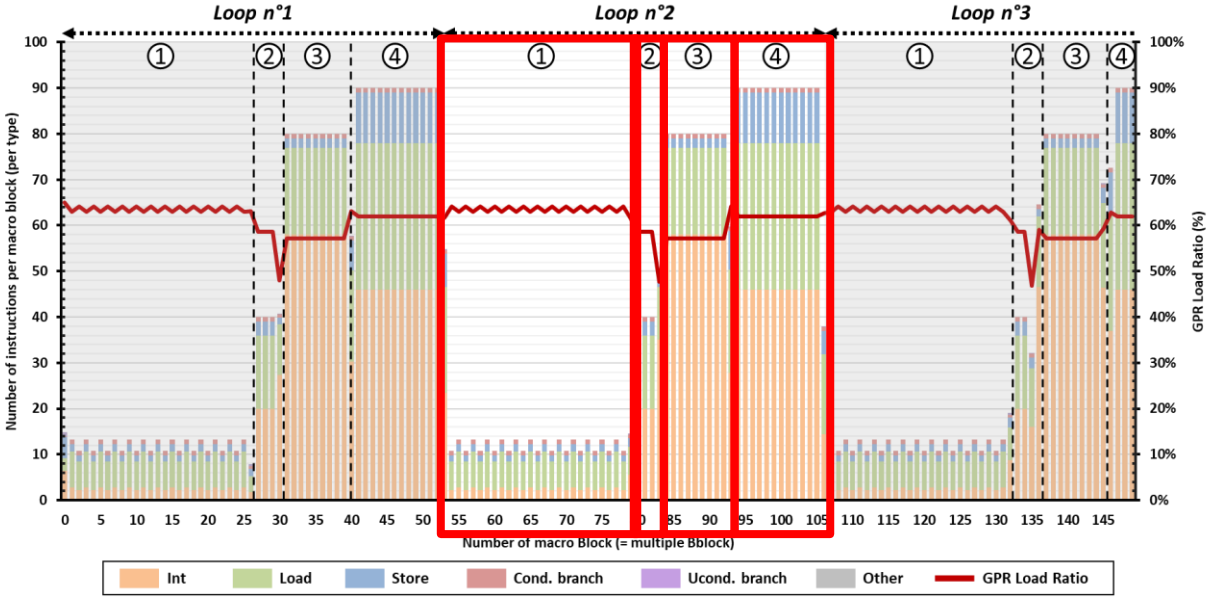
In-Depth Analyses
→ 1st laboratory validation

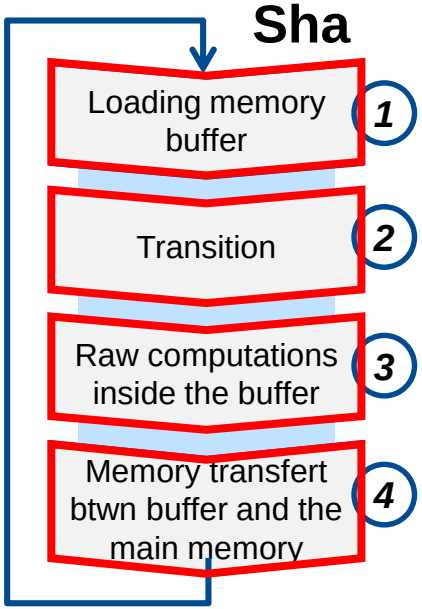
Test platform



Use in real-World conditions
→ 2nd validation under beam

Profiling tool : Example of software profiling



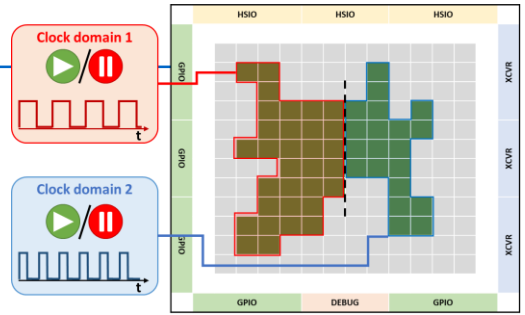


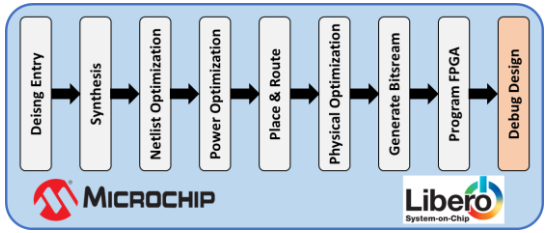
Identify the 4 computation steps of the Sha_o0 algorithm

Emulation platform : SmartDebug

Main characteristics of SmartDebug tool by Microchip:

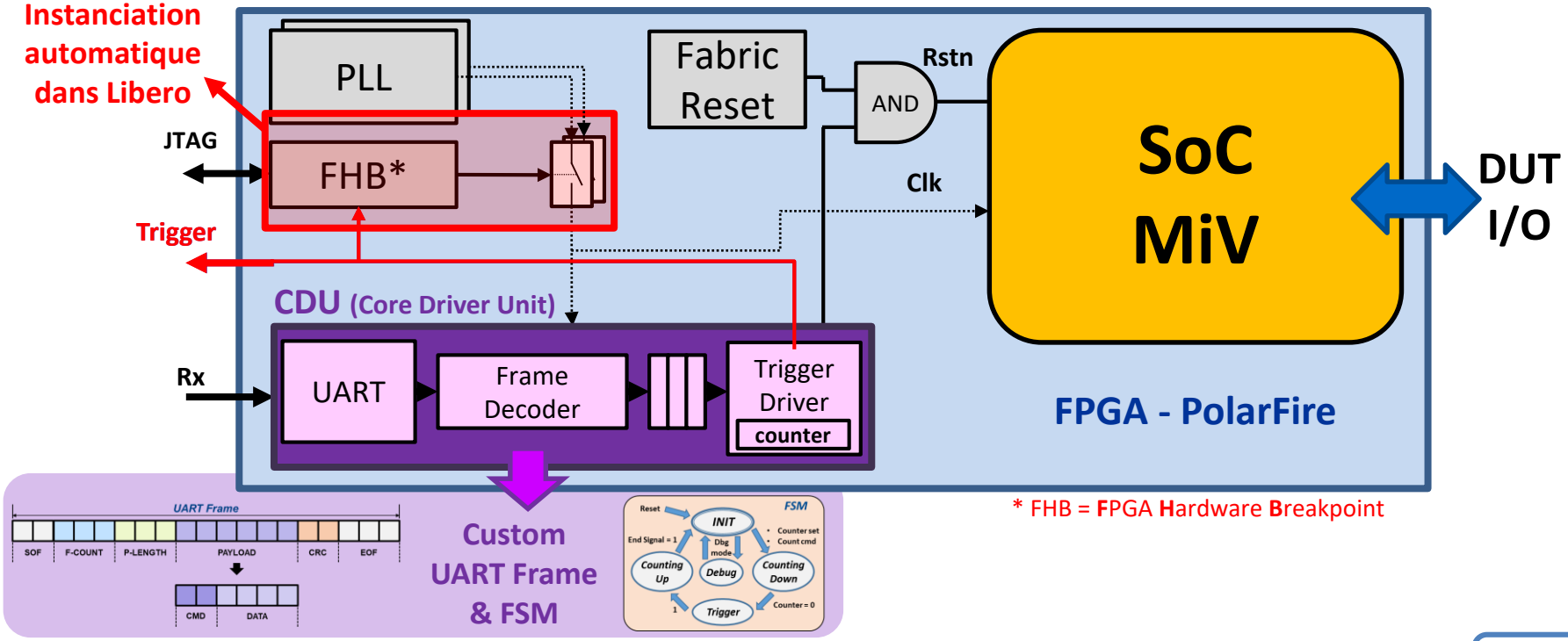
- Start / Stop the clock in each clock domain (can be triggered by an internal signal)
- R/W each DFF / memory inside the FPGA (JTAG)
- Can be scripted by TCL functions
- Integrated inside the design flow environment (Libero)





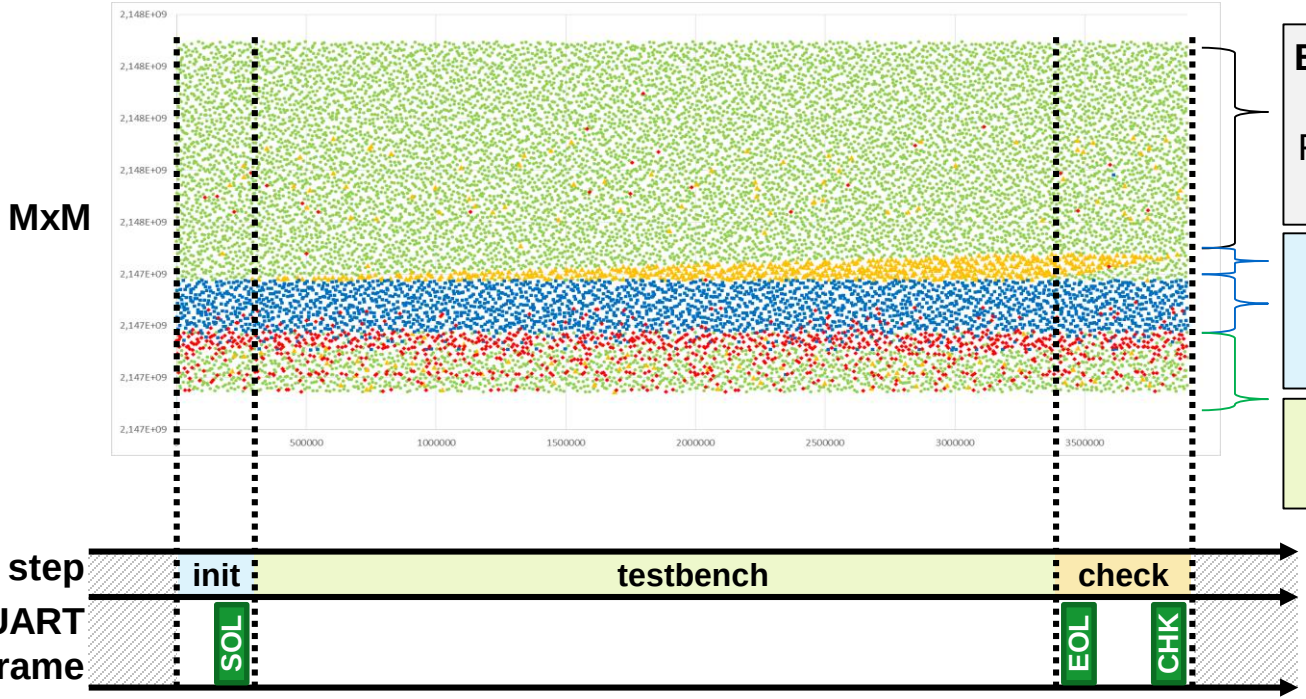
Need to create a trigger signal for the fault injection and a list of faults

Emulation platform : Fault injection



* FHB = **F**PGA **H**ardware **B**reakpoint

Emulation platform : Example of FI results in MM of MxM



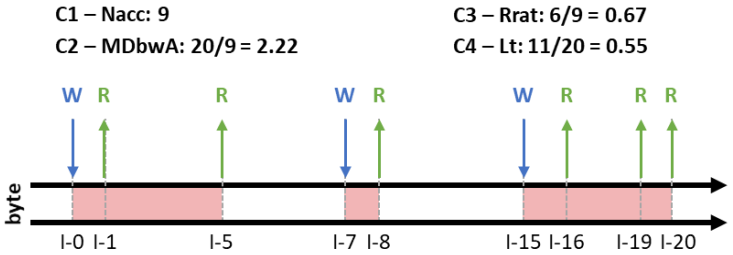
BSS + Heap + Stack sections
Mainly **No Impact**
Few **T-SDC** & **SEFI** depending of the computation

Data section
T-SDC on resulting matrix
P-SDC on data matrixes

Text section
SEFI on executed code

“Proof of concept” case study

Profiling data

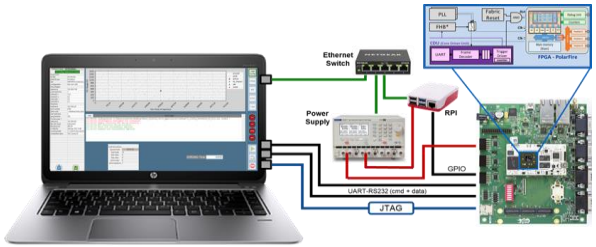


→ *Custom script to analyze Spike exec. traces (inst. accurate)*

6 memory sections:

- 5 for Main Memory (MM) (text, data, bss, heap, stack)
- 1 for General Purpose Registers (GPR)

Fault tolerance data



[Noizette_2023]

→ *Custom fault injection tool using debugging interface*

Statistical Fault Injection (SFI):

- Error margin
- Confidence level

[Leveugle_2009]

➡ **Reproduce each step / key element of the methodology**

SUMMARY

1. INTRODUCTIONS & MOTIVATIONS
2. PROPOSED METHODOLOGY & CASE STUDY
3. **EXPERIMENTAL RESULTS**
4. CONCLUSION AND PERSPECTIVES

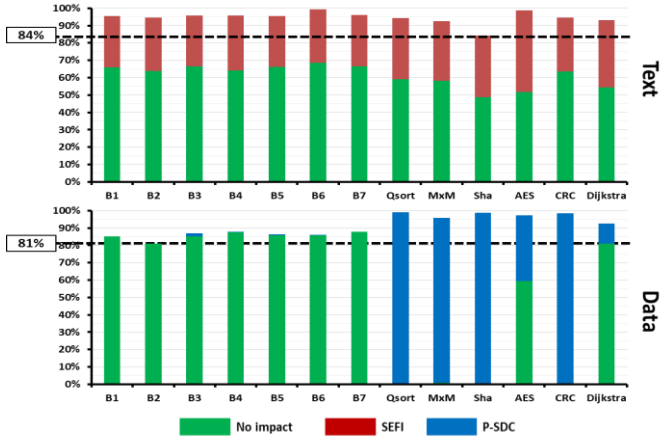
Application in Main Memory (MM) with o0 (1/2)

Sep. SFI in 5 MM sections ONLY for training programs → Computing the $Rf_{Si,Mi}$ and $\overline{Rf_{Si}}$ for the training programs → Compute the Er of all the SW (training + test) w/ their profile → Obtain the Mr by SFI in MM of all programs (training + test)

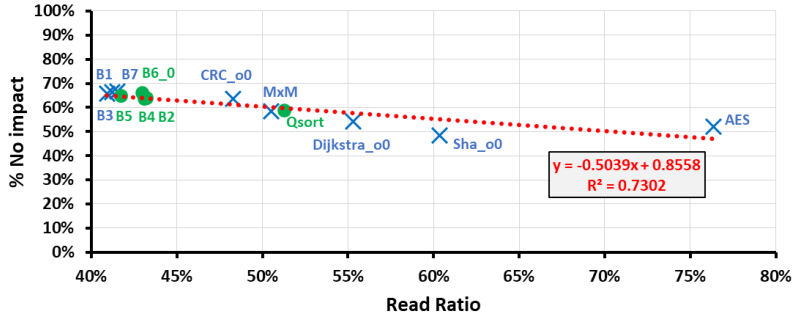
"Training" phase

"Verification" phase

Faults in Text and Data mem. sections



Regression formula of Lt profiling metric in text memory section $(Rf_{text,Lt})$

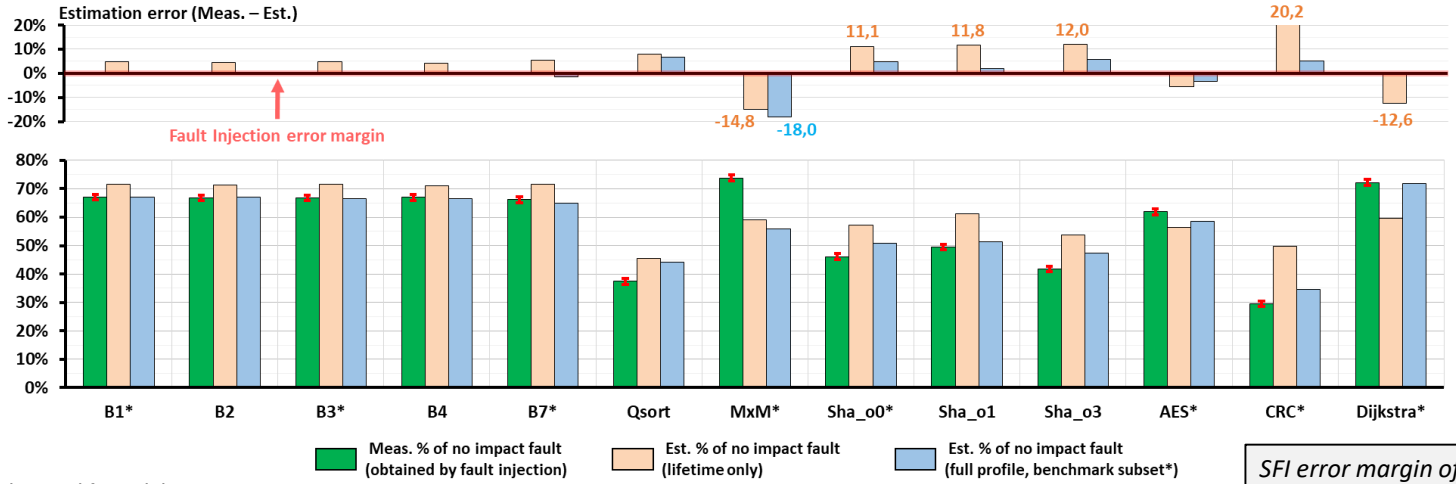


SFI error margin of 3% + confidence level of 95% = 1068 faults per mem. section per benchmark

Er = Estimated Robustness, Mr = Measured Robustness

SFI = Statistical Fault Injection

Application of the methodology in MM / o0 (2/2)



* = Benchmarks used for training

SFI error margin of 3% + confidence level of 95% = 23874 faults injected per benchmarks

Comments:

- The error in estimating the robustness is below 5% with respect to the SFI
- The full profiling improves the estimation of the robustness compare to the one only based on Lt

Application of the methodology in GPR / o0

SFI in GPR section ONLY for training programs → Computing the $Rf_{GPR,Mi}$ and $\overline{Rf_{GPR}}$ for the training programs

"Training" phase

Compute the Er of all the SW (training + test) w/ their profile → Obtain the Mr by SFI in GPR of all programs (training + test)

"Verification" phase

Program	Measured robustness	Difference (our method, lifetime only)	Difference (our method, four metrics)
B6	90.8 %	9.2%	-2.5 %
Sha*	91.4 %	0.0 %	0.0 %
CRC*	89.7 %	0.0 %	0.0 %
Dijkstra	88.5 %	3.8%	-0.3 %

Programs are compiled in o0

* = Benchmarks used for training

SFI error margin of 1% + confidence level of 99.8% = 22000 faults injected per benchmark

Comments:

- The % of no-impact is higher in GPR (~ 90%) than MM (~ 30-70%)
- The error in estimating the robustness is below 5% with respect to the SFI

Er = Estimated Robustness, Mr = Measured Robustness

SFI = Statistical Fault Injection

Impact of compilation options (1/2)

SFI = Statistical Fault Injection

Er = Estimated Robustness, Mr = Measured Robustness

SFI in GPR section ONLY for training programs (o0,o1,o3)

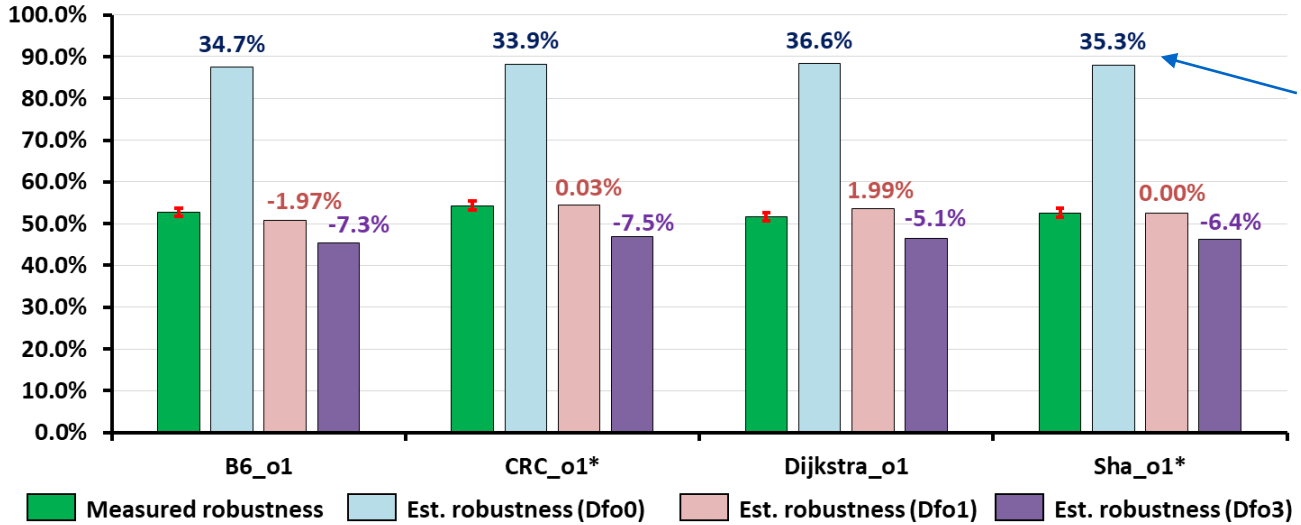
Computing the $Rf_{GPR,Mi}$ and $\overline{Rf_{GPR}}$ for the training programs

Compute the Er of all the SW (training + test) w/ their profile

Obtain the Mr by SFI in GPR of all programs (training + test) (o1)

"Training" phase

"Verification" phase



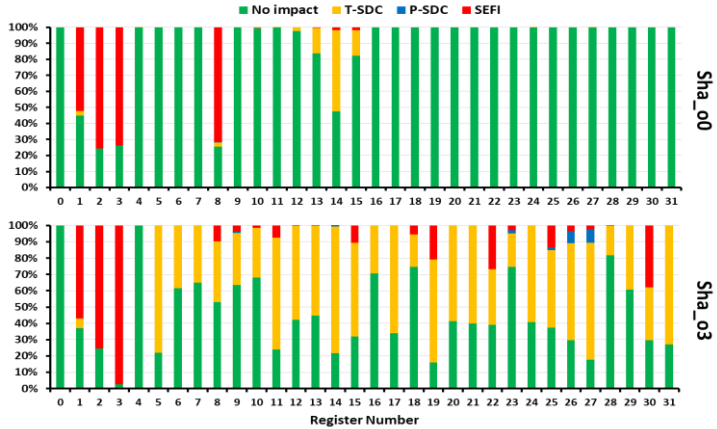
Estimation error compared to the measured robustness

SFI error margin of 1% + confidence level of 99.8% = 22000 faults injected per benchmark

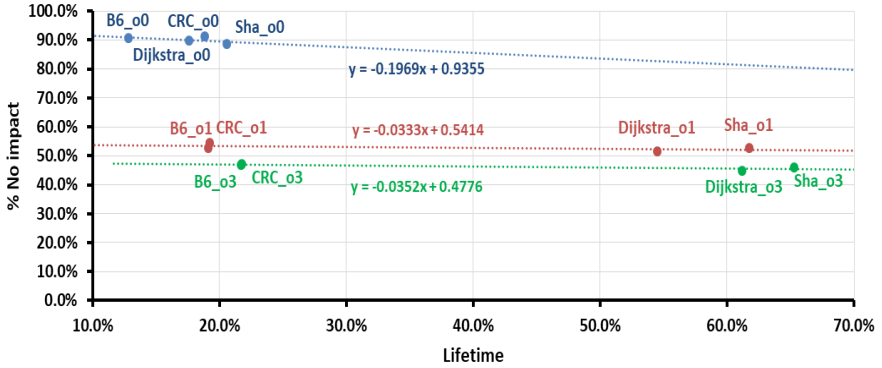
* = Benchmarks used for training

Impact of compilation options (2/2)

Fault repartition in GPR for Sha (o0 & o3)



Regression formula of Lt profiling metric in GPR memory section ($Rf_{GPR,Lt}$)



Comments:

- Regs n°1 (ra), n°2 (sp), n°3 (gp) are prone to SEFI where regs n°12-15 (a2-a5) are more prone to T-SDC
- No significant change has been observed in the MM (hyp. Test programs are too small)

How to validate the methodology with a radiation beam ?

Computation of the Est. dynamic Xs



$$Xs_{static} = Xs_{ref} \times Cs$$

Neutron Xs of LSRAM blocks (available from Microchip test report)

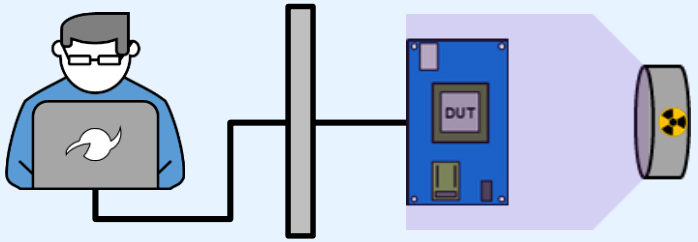
Code size of the software

$$Xs_{dyn} = Df_{SW} \times Xs_{static}$$

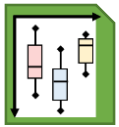
Complete version *(all profiling metric were used)*

«Degraded» version *(only the Lt metric was used)*

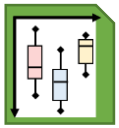
Measurement of the Exp. dynamic Xs



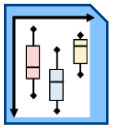
$$Xs_{dyn} = \frac{\#T_{SDC} + \#P_{SDC} + \#SEFI}{fluence}$$



Est. Dynamic Xs



?



Exp. Dynamic Xs
(reference)

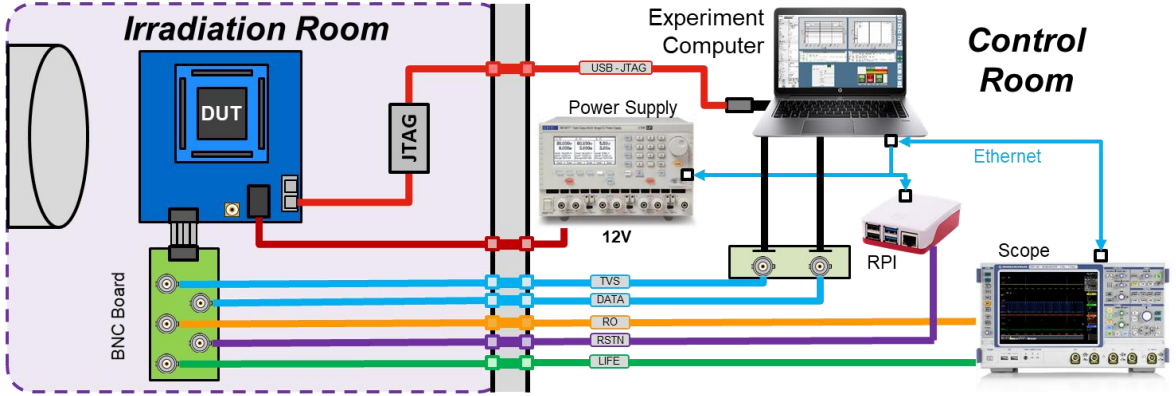
Accelerated Beam Experiment (1/2)

Facility & Experiment:

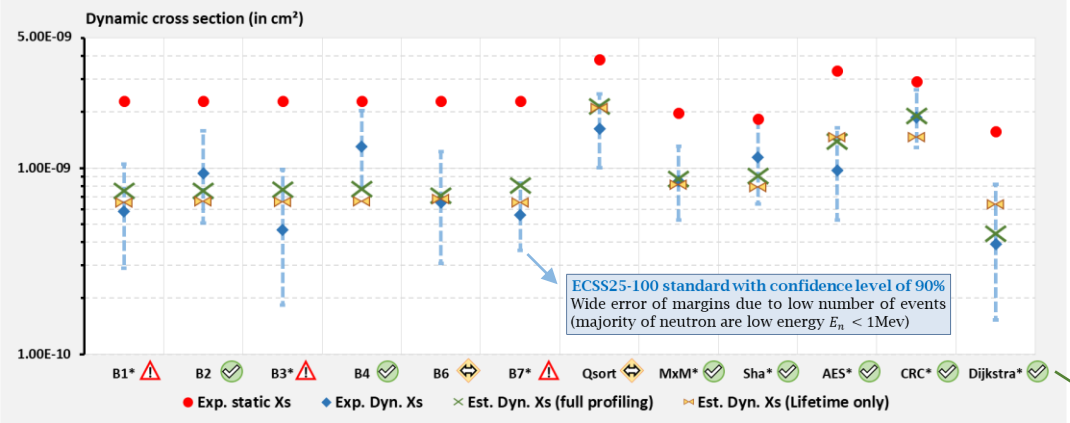
- December 2023
- ISIS (ChipIR) Rutherford Appleton Lab., UK
- Atmospheric-like neutron spectrum (x10^9)







Accelerated Beam Experiment (2/2)



Benchmark	Code Size (kB)	Runtime (ms)	Fluence (n/cm²)	Xs_{static} (cm²)	$Xs_{dynamic}$ (cm²)
B1	19,7	77,2	1.37×10^{10}	2.30×10^{-9}	5.84×10^{-10}
B2		70,3	1.06×10^{10}		9.34×10^{-10}
B3		78,4	1.08×10^{10}		4.67×10^{-10}
B4		71,2	1.07×10^{10}		1.31×10^{-9}
B6		71,6	1.07×10^{10}		6.54×10^{-10}
B7		117,4	3.21×10^{10}		5.60×10^{-10}
Qsort	33,1	250,6	9.21×10^{9}	3.84×10^{-9}	1.63×10^{-9}
MxM	17,1	77,9	1.76×10^{10}	1.98×10^{-9}	8.52×10^{-10}
Sha	15,8	101,0	9.64×10^{10}	1.84×10^{-9}	1.14×10^{-9}
AES	28,7	85,5	1.03×10^{10}	3.34×10^{-9}	9.73×10^{-10}
CRC	25,0	65,2	1.28×10^{10}	2.91×10^{-9}	1.87×10^{-9}
Dijkstra	13,5	71,7	1.29×10^{10}	1.57×10^{-9}	3.89×10^{-10}

The Dyn. Xs estimated in "full profiling" version is the closer to the Exp. Dyn. Xs.

Comments:

- All estimated Xs computed in "full profiling" version are within experimental error margin
- The "full profiling" version is better than or equivalent to the "Lifetime only" one in 75% of the cases

SUMMARY

1. INTRODUCTION & MOTIVATIONS
2. PROPOSED METHODOLOGY
3. EXPERIMENT SETUP AND RESULTS
4. **CONCLUSION AND PERSPECTIVES**

Conclusion & Perspectives (1/2)

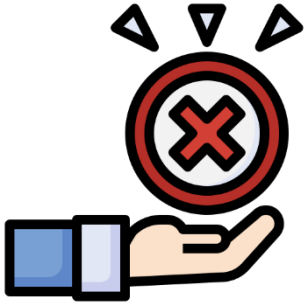
Main results:

- ✓ Proposition of a methodology for early fault tolerance assessment:
  - ↳ Based only on high-level profiling data
  - ↳ Suitable for test & dev. teams throughout the whole design flow
- ✓ “Proof of concept” case study has been validated:
  - ↳ with fault injection (importance of using the good compilation option)
  - ↳ under real neutron beam



Current limits:

- Only 4 memory profiling metrics are considered (Nacc, Rrat, MDbwA, Lt)
- Test on 1 simple micro-architecture (no cache, in-order)
- Limited number of benchmarks



Conclusion & Perspectives (2/2)

Perspectives:

- ❑ Test the use of non-linear regression formulas to calculate the Er_{SW}
- ❑ Add new profiling metrics to cover more HW and SW mechanisms
- ❑ Use others complex components with other ISA (ARM, Power, SPARC with LEON)
- ❑ Add more programs to the currently reduced set

➡ **Validated "Proof of concept" that paves the way to confirm and generalize the use of the proposed methodology**



Thank you
for you attention



Luc Noizette, PhD
Radiation Engineer
Digital Electronic Hardening

3 avenue du Hoggar – Les Ulis – CS 70117
91978 Courtaboeuf Cedex – France
Phone: +33 6 80 85 12 67
Mail : lnoizette@nucletudes.com
www.nucletudes.com

Do you have any questions ?

References

[James_2021] B. James et al., "Investigating How Software Characteristics Impact the Effectiveness of Automated Software Fault Tolerance" in IEEE Transactions on Nuclear Science (TNS), vol. 68, no. 5, pp. 1014-1022, **2021**

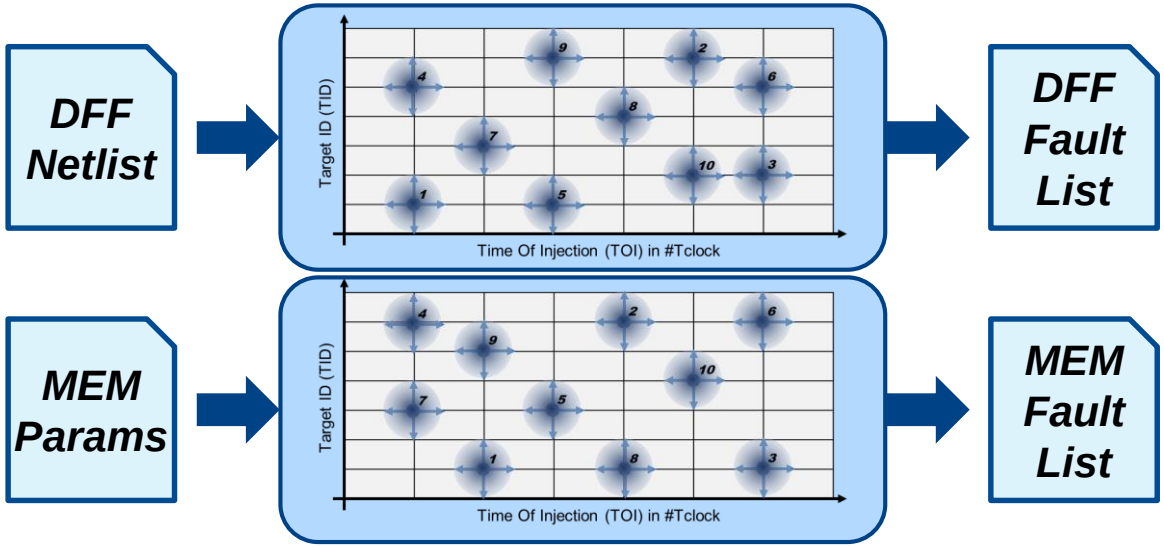
[Guthaus_2001] M. R. Guthaus et al., "MiBench: A free, commercially representative embedded benchmark suite" in Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization pp. 3-14, **2001**

[Leveugle_2009] R. Leveugle et al., "Statistical fault injection: Quantified error and confidence" in Design, Automation & Test in Europe (DATE) Conference & Exhibition, pp. 502-506, April **2009**

[Terrile_2015] R. J. Terrile et al., "Calibrating the Technology Readiness Level (TRL) scale using NASA mission data" in IEEE Aerospace Conference, pp. 4036-4044, March **2015**

[Cazzaniga_2021] C. Cazzaniga et al., "First Tests of a New Facility for Device-Level, Board-Level and System-Level Neutron Irradiation of Microelectronics" in IEEE Transactions on Emerging Topics in Computing (TETC), vol. 9, no. 1, pp. 104-108, March **2021**

[Noizette_2023] L. Noizette et al., "Understanding the Link Between Complex Digital Devices Soft Error Rate and the Running Software" in IEEE Transactions on Nuclear Science, vol. 70, no. 8, pp. 1747-1754, 2023

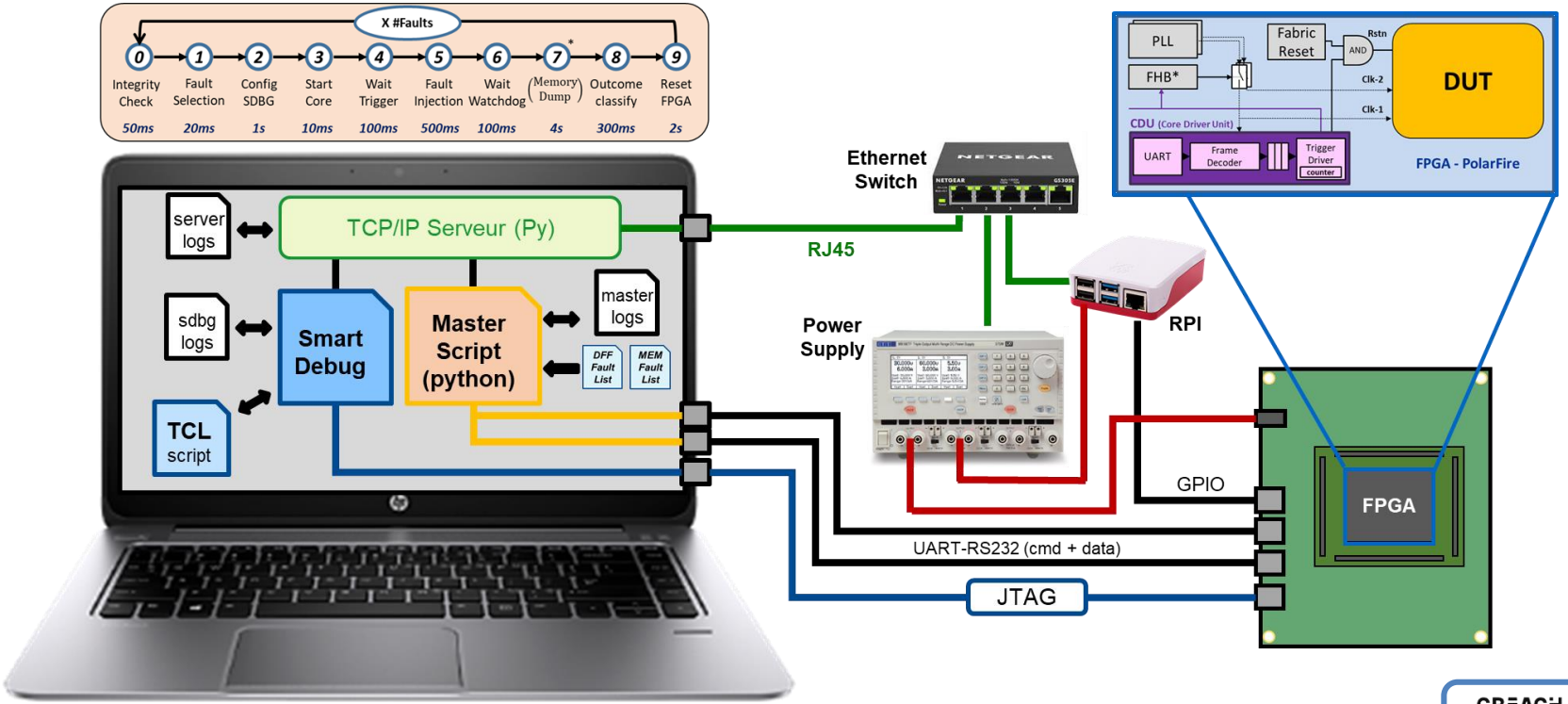


Specifications:

- Pseudo-Random (seeds)
- Repeatable (split into seq)
- Uniform distribution
- Statistical number
- DFF & Memory

Delegated to another script (python, …)

Overview of the FI Tool



Computation of the Xs

$$Xs_{static} = Xs_{ref} \times Cs$$

Neutron Xs of LSRAM blocks given by manufacturer

Code size of the benchmark

Experimental

$$Xs_{dynamic} = \frac{\#T_{SDC} + \#P_{SDC} + \#SEFI}{fluence}$$

Number of SEFI events observed during one run

Estimated

$$Xs_{dynamic} = Er_{SW} \times Xs_{static}$$